Scott A. Kamber (*pro hac vice*)
skamber@kamberlaw.com
David A. Stampley (*pro hac vice*)
dstampley@kamberlaw.com
KamberLaw, LLC
100 Wall Street, 23rd Floor
New York, New York 10005
Telephone: (212) 920-3072
Facsimile: (212) 202-6364
Interim Class Counsel

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN JOSE DIVISION

IN RE IPHONE APPLICATION LITIG.	CASE NO. 5:11-MD-02250-LHK The Honorable Lucy H. Koh **DECLARATION OF MANUEL EGELE IN SUPPORT OF PLAINTIFFS' MOTION FOR CLASS CERTIFICATION**

TABLE OF CONTENTS

A. Overview of opinions 3

B. Qualifications 4

C. Application life-cycle 5

1. Functions and APIs 5

2. General iOS and iDevice system information 6

3. iOS updates and supported hardware 7

4. The software development kit 8

5. Writing an iOS application (app) 8

6. Submitting an app for review & the review process 10

7. Purchasing, installing, and executing apps 12

8. Device identifiers 13

D. Research results 17

1. Advertising and statistics libraries 18

2. PiOS results 19

E. Resource consumption 20

F. Other information 23

G. Endnotes 23

1. I, Manuel Egele, declare as follows: I am Manuel Egele, Dr. techn. (equiv. PhD). Currently, I am a post-doctoral researcher at Carnegie Mellon University. I submit this declaration in support of Plaintiffs' Motion for Class Certification in this matter. I am familiar with and have personal knowledge of the matters set forth in this declaration and if called upon to do so, could and would testify competently thereto, except where my knowledge is based upon information and belief, and as to those matters, I understand and believe them to be true.

A. Overview of opinions

2. I was asked to give my opinion regarding the effect on the iPhone user population of certain technology features of iPhones and software that interacts with iPhones including transmittal of user and geo-location information to Apple and/or other parties.

3. It is my overall opinion that Apple does not implement reasonable and effective mechanisms that protect the privacy of iDevice users. My research shows that the majority of third-party applications access and transmit privacy sensitive information to other Internet-connected devices (e.g., servers).

4. Instead of providing effective technical privacy preserving alternatives, Apple relies on the program license agreement (PLA) to safeguard privacy sensitive information from the data-stores of third-party application developers. While the app review process seems to be reasonably effective in preventing destructive applications from entering the app store, access and transmission of privacy sensitive data by third-party applications is commonplace and is only barely subjected to enforcement by Apple.

5. Because Apple does not enforce reasonable privacy measures in the application review process, iDevice users cannot expect their privacy being safeguarded from privacy-invading third-party applications. The alternative for privacy-aware users would be to assess themselves whether applications pose a privacy risk. However, regular users do not possess the skills and knowledge that is necessary to perform such analyses. Furthermore, Apple does not expose the necessary information for users to make educated decisions with respect to privacy considerations.

6. By finally deprecating the use of the UDID in iOS 5 Apple seemed on the right track to regain user's confidence in Apple's commitment towards their privacy. Unfortunately, Apple implemented the new advertisingIdentifier in iOS 6, knowing that it has the very same adverse effects on privacy as the deprecated UDID system.

7. Furthermore, by not distinguishing between library code and application code, Apple enables advertisers and marketers to make use of privileged system resources (e.g., location information). These activities consume energy and thus reduce the time until a recharge is necessary, as well as the overall lifetime of the battery. Additionally, auxiliary data transmissions that are not explicitly authorized by the user still consume bandwidth.

B. Qualifications

8. My highest academic title is Dr. techn., the Austrian equivalent to a PhD. Furthermore I hold a Dipl.-Ing., the Austrian equivalent to a Master's degree. I earned both degrees at the Vienna University of Technology in the years 2006 and 2011 respectively.

9. I have been active in the field of system security since 2005. My expertise includes static and dynamic analysis of mobile and desktop applications, web-security, and privacy considerations in mobile systems and online social networks.

10. I have been the lead author and researcher of multiple peer reviewed scientific articles, including PiOS [1], a static analysis system that detects privacy violations in third-party iOS applications. I was honored to receive a distinguished paper award for the research that lead to the PiOS system. (See also section D.) Beyond PiOS I published several other papers on mobile and systems security topics.

11. I was the co-author of a successful research grant proposal, "DarkDroid: Exposing the Dark Side of Android Marketplaces." This resulted in the award of a US$ 2.5M grant funded by Defense Advanced Research Projects Agency (DARPA) for research starting January 2012.

12. My full curriculum vitae is attached and lists additional qualifications, as well as all scientific publications I authored in the last ten years.

C. Application life-cycle

13. This section is intended to cover the life-cycle of an application. I will explain the steps leading from the creation (i.e., writing the source code) of an application to the execution by the Apple iOS device to the best of my understanding. Furthermore, I will cover necessary background materials as they are prerequisites to understand any opinions contained in this document.

1. Functions and APIs

14. Functions in programs are abstraction mechanisms that allow the developer to reuse the same functionality in different contexts without duplicating the code. A function has a name, a return value, a set of formal parameters, and a function body implementing the functionality. A function declaration (or prototype), consists of the same information but omitting the implementation. Thus, for a programmer to make use of a function, the declaration provides all the information needed. That is, the concrete implementation of a function is abstracted.

15. The declaration describes an interface the developer can rely on to make use of the function. For example, the function (`add`) implementing the addition of two integers (`x` and `y`) can be declared as:

```
int add (int x, int y);
```

This can be interpreted as there is a function with the name add, this function returns a result of type int. Furthermore, the function receives two arguments (or parameters). Both arguments are of type int, one will be known under the name x the other as y. Of course, the trivial implementation of add is as follows:

```
int add(int x, int y) { return x + y; }
```

That is, the function add will take the first argument `x` and add it to the second argument `y`. Then add will return the result of this computation as its result.

16. Functions, with related functionality are commonly combined into modules or libraries. For example, together with the above mentioned add function, we can analogously define the remaining three basic arithmetic operations as `sub`, `mul`, and `div`.

17. Commonly, function definitions are collected in separate so-called header files. The reason is that the declarations alone are sufficient for a developer to make use of functions, while the concrete implementations can be distributed in binary only form (i.e., without source code). That is, a developer can write code that makes use of functions by only knowing the declarations. Of course, the implementation of the functions (commonly contained in libraries) need to be present when the code is executed.

18. Furthermore, logically coherent function declarations and implementations are commonly combined into so-called *Application Programming Interfaces* (APIs). Our above example can therefore implement a basic arithmetic operations API. Similarly, APIs that provide functionality such as network communication, file system access, or html rendering (i.e., the display of a website) are commonly available on modern systems, including Windows, OS X, and iOS.

19. Apple combines such APIs with additional resources, such as images or configuration settings into so-called *frameworks*. A non-exhaustive list of frameworks available on all iOS devices contains, for example, the *CoreLocation* framework which provides functionality that allows an application to obtain the current position of the device. The *UIKit* framework provides access to a variety of device relevant information, such as a unique device identifier (UDID). As a last example I want to mention the *AddressBook* framework which provides functionality to read, write, and modify contents stored in the device's address book.

20. It appears that the APIs Apple documents on its developer website [1] are those that third party developers are allowed to make use of when creating iOS applications. However, additional restrictions might apply such as those set forth in the program licensing agreement as discussed later on in this document.

2. General iOS and iDevice system information

21. Apple iOS is the operating system that Apple installs on their iPhone, iPad, and iPod touch mobile devices (i.e., iDevices). In fact, iOS is the only Apple sanctioned operating

[1] http://developer.apple.com/library/ios/

system supported on iDevices, and iDevices are the only devices that support iOS. Thus, iOS and iDevices are always encountered together.

22. The iDevices themselves contain CPUs that implement different versions of the ARM instruction set architecture. The iOS operating system is a Unix style system that shares code and/or functionality with Apple's Darwin kernel. Furthermore, all iOS devices contain the so-called Objective-C runtime which provides support functions and data structures needed by all applications installed on such iDevices.

23. The operating system manages the hardware resources (e.g., the wireless network interface or internal storage) and provides standardized interfaces for applications to communicate with these resources (e.g., open a connection to a remote host that is reachable on the Internet, or permanently store a file to internal storage).

24. Furthermore, it is the operating system's responsibility to provide strong isolation between different applications. It is this isolation mechanism that prevents a potentially malicious application from accessing sensitive information managed by another application (e.g., a user password). Many other security mechanisms are also built into the operating system.

25. It is worth noting that due to the privileged nature of the operating system, no application can access hardware directly. All accesses to hardware resources are without exception mediated by the operating system.

3. iOS updates and supported hardware

26. Apple releases from time to time updated versions of their iOS operating system. However, these updates are only made available for devices that fulfill the hardware requirements inherent to the iOS version. That is, devices whose hardware is not powerful enough to run the latest version of iOS, cannot be updated. Their users, thus, cannot benefit from the changes and improvements of that iOS version. For example, it is my understanding that the current (as of this writing) iOS version 6 cannot be installed on iPhone 3 devices. However, whenever Apple releases an update to iOS, users with supported devices can install this

version seamlessly via iTunes or in more recent iOS versions wirelessly. Regular users can only install the publicly released versions of iOS.

4. The software development kit

27. To facilitate the creation of third-party applications, Apple distributes the iOS Software Development Kit (SDK). The SDK contains resources needed to author and test applications that can be executed on iDevices. Among other resources the SDK includes all frameworks (see C.1) that the developer can expect to be present on the target iOS device.

28. To the best of my knowledge Apple releases an updated version of the SDK with each version of iOS. Thus, there is a correspondence between iOS and SDK version.

29. Furthermore, the iOS SDK contains not only the latest versions of frameworks but also a series of older versions that correspond to previous iOS versions. This allows the developer to create applications that are backwards compatible with older version of iOS.

5. Writing an iOS application (app)

30. A developer who wishes to create an iOS application (app) starts her endeavor by obtaining an Apple computer and the XCode development environment. XCode can be downloaded free of charge from the Apple website. Subsequently, the developer downloads, also free of charge, the iOS SDK from the Apple website. Once installed the developer can author iOS applications at will.

31. I will now illustrate how a developer accesses different kinds of user or device data.

a. Unique device identifier (UDID)

The UDID is a string of 40 hexadecimal characters that uniquely identifies any iOS device. Furthermore, the UDID is constant, and thus cannot be changed or altered by the user of the device. Therefore, the UDID can be used to uniquely identify a given device during its lifetime. A developer accesses the UDID by adding the following code into her application.

```
UDID = [[UIDevice currentDevice] uniqueIdentifier];
```

More precisely, the developer calls functionality in the above-mentioned *UIKit* framework which is installed by default on all iOS devices. The developer can now make use of the UDID as she wishes. For example, she can provide account-less login to a web-service that works in tandem with the application. Alternatively, or additionally, the developer is at liberty to also transmit this information to a server that she manages to aggregate an application usage profile. For example, if this code is included in the startup code of an application, the developer can track the date, time, and UDID of devices on which her application is launched.

According to Apple documentation [2] the UDID property first appeared with iOS 2.0 and was deprecated with iOS version 5.0. Note that deprecating an API or property does not mean it is no longer available, it merely indicates that developers should refrain from using it as it will probably go away in the future.

b. Current location information

As opposed to the UDID, an application cannot directly retrieve the current location. Instead the application asks the CoreLocation framework to be informed of the current position. To this end, the application creates a so-called location-manager (i.e., an object of type CLLocationManager) and invokes the `startUpdatingLocation` method to be notified of the current location or location changes. The CoreLocation framework then obtains the current location with the help of the operating system. If necessary the GPS receiver is activated to obtain the current geo-location. Alternatively, CoreLocation can also obtain a coarser-grained location from a variety of sources such as the mobile operator cell-id to which the phone is currently connected to, or wireless network information. Once CoreLocation obtained the required information it will relay this information to the above-mentioned location-manager by invoking its `location-`

---

[2] http://developer.apple.com/library/ios/#documentation/uikit/reference/UIDevice_Class/DeprecationAppendix/AppendixADeprecatedAPI.html#//apple_ref/occ/instp/UIDevice/uniqueIdentifier

`Manager:didUpdateLocations:` method. Prior to iOS 6 CoreLocation would insead call `locationManager:didUpdateToLocation:fromLocation:`.

c. Address book contents

The developer can copy the contents of the address book by adding the following code to the program.

```
ABAddressBookRef addressBook =
ABAddressBookCreate();
NSArray *people = (NSArray *)
ABAddressBookCopyArrayOfAllPeople(addressBook);
```

Until iOS 6 access to the address book was not mediated at all. That is, every application was able to copy the address book without the user noticing. With iOS 6 Apple changed this privacy nightmare by mandating from developers that the user is informed about such accesses (via an authorization dialog) and user-consent is obtained.

32. While the developer authors the application's source code in simple text files, this is not directly executed by the iOS device. The CPU in an iOS device only understands binary data that conforms to the ARM instruction set. Therefore, the developer will make use of a compiler to translate her source code into the binary machine code that can be executed on the device. This compilation entails a series of consequences that become relevant for the application review process that we will discuss next.

6. Submitting an app for review & the review process

33. Once a developer deems her application fit for use by the general public, she submits it for review to the Apple app store. However, prior to submission the developer needs to register with Apple. To this end, she pays the US$ 99 annual fee and agrees to the so-called iOS developer program license agreement (PLA).

34. The PLA, among other things, sets forth a set of rules that third-party developers have to abide by before they can have their applications published on the app store.

35. The PLA has been heavily revised by Apple over time. However, historically a section titled "User Interface, Data Collection, Local Laws and Privacy" regulates how applications can make use of user and device data.

36. Once the developer paid the fee and agreed to the PLA she can obtain cryptographic signing keys from Apple. Before submitting an application for review the developer signs the application with those keys. This allows Apple to verify that the application was submitted by the developer in question and not some other unrelated party.

37. Subsequently, the application will go through the application review process. Unfortunately, the details of this process are one of the best kept secrets in the iOS ecosystem. However, technically there are two broad classes of analyses that Apple can perform on an application during this process.

38. First, *dynamic analysis* refers to all techniques that monitor the effects of an application during execution. For example, capturing and analyzing the network traffic an application is producing is considered dynamic analysis.

39. Second, *static analysis* techniques are those that reason about the application without actually executing it. For example, analyzing all API calls an application can make is only possible statically. Forensic analysis inspects the static artifacts of system execution and thus can be classified as static analysis.

40. However, when using the above-mentioned network traffic example again, we have to assert the following: Because the transmission of data over the network does not leave forensic evidence on the iDevice's file-system, forensic analysis of the file-system is not an adequate method to assess what data if any was transmitted.

41. That said, both, static and dynamic analysis techniques have advantages and disadvantages. One of the purposes of the application review process is to assess whether an application adheres to the rules set forth in the program license agreement (PLA). Should the application violate any of the rules, Apple will notify the developer of the identified issue and the developer can modify the application and resubmit a modified version.

42. If the application passes the review process it will be made available on the app store either free of charge or for a fee as determined by the developer.

43. Should the developer want to make changes to her application (e.g., adding a new feature) she has to resubmit the modified application and Apple will perform another round of review before the modified version is published on the app store.

44. According to Mr. Shoemaker [4][p33,1-3] Apple is conducting dynamic analysis by means of network traffic monitoring ("capturing all of the web traffic that the application sends to the backend server and") for a subset of the reviewed applications during a so-called *technical investigation*. Additionally, Apple makes use of static analysis techniques during their application review process.

7. Purchasing, installing, and executing apps

45. Once available on the app store, users can purchase applications from the app store. Of course, free applications can be downloaded without payment.

46. The user can perform the necessary steps from iTunes and synchronize the downloaded application to her iDevice. Alternatively, the user can download the app directly from the AppStore application on the iDevice itself. Once installation is complete, the user will have a new icon on her device's screen that allows her to launch the newly installed application.

47. Upon application launch the operating system will start a new process for this application. Referring back to our example of accessing the UDID, the binary code of the application will call into the UIKit framework to obtain the UDID. The UIKit framework is included and installed alongside with iOS. Thus, all applications that want to access the UDID access it in the same way. Note that [8] illustrates four possibilities to access the UDID. However, only the above stated method (i.e., `[[UIDevice currentDevice] uniqueIdentifier]` is considered API by that same document).

48. Regardless of what method is used to access the UDID, it is always the same persistent unique identifier that will be returned to these function calls.

(1) Accessing sensitive resources

49. Note that access to some security and privacy relevant resources is monitored or restricted during runtime (i.e., while the application is executing). For example, accessing location information as illustrated above will, at runtime, trigger a dialog box where the user is prompted for consent as to whether the current application can access location information.

50. Starting with iOS 6 a similar dialog will be shown when an application tries to access address book contents. Prior to iOS 6 however, access to the address book was not mediated at all. That is, each application could read and write to the address book.

51. While such user prompts are a first step towards helping users understand an application's intentions, the protection provided by such mechanisms are negligible. That is, there is no technical mechanism in place that enforces that data is only used for the stated purpose it was gathered for. This is especially problematic in the light of third-party libraries as we will see later on.

8. Device identifiers

[redacted]

[redacted]

[redacted]

58. Strictly speaking the UDID is a device identifier. However, the pervasive use of this identifier makes it a prime candidate to aggregate it with additional privacy sensitive, and personally identifiable information. Thus, once such aggregation is performed, the UDID itself becomes personally identifiable. This view is supported by Mr. Neuenschwander's deposition [3][p88 19 – p89 3] "…and then a developer who, you know, calls the API to get that identifier from iOS and subsequently puts it into – links it to personal details, then the – or at least Apple would consider, then, that because that identifier is linked to the personal details, that that identifier would, in that case or that situation, be – should be treated as personal itself".

[redacted]

60. Similarly, any application can aggregate personal data via the UDID. For example, it is trivial for a social networking app to correlate the UDID with the social network ac-

count and all information associated with that account (e.g., real name, date of birth, etc.). Apple is aware that the Facebook app accesses the UDID along with address book contents [13]. "In this case, the app, Facebook, is using both the Address Book apis to access Contacts information and it is also accessing the UDID for the device". Of course, an outside observer cannot assess whether Facebook uses the UDID to aggregate data in their system. However, once the UDID is transmitted to their systems nothing is stopping them.

[redacted]

[redacted]

D. Research results

65. The personal data stored on mobile devices deserves protection from preying third-party applications. Thus, from fall 2009 to summer 2010 I conducted a research project to assess how third-party iOS apps interact with privacy sensitive information available on iOS devices. During that project I implemented the PiOS analysis system [1]. This work was published at the highly reputable Network and Distributed Systems Security Symposium (NDSS). NDSS implements a rigorous peer review process to determine the papers that are added to the proceedings each year. Moreover, despite the many high quality submissions to NDSS the program committee decided to award PiOS with a distinguished paper award.

66. PiOS performs automated static analysis on iOS applications. To this end, PiOS analyzes a series of apps in sequence. More specifically, PiOS first reconstructs the so-called control-flow-graph for each app. Based on the control-flow graph PiOS then performs static data-flow analysis to determine what privacy sensitive information is accessed by an application and whether that information is sent to the Internet.

---

[3] http://developer.apple.com/library/ios/#documentation/AdSupport/Reference/ASIdentifierManager_Ref/ASIdentifierManager.html#//apple_ref/occ/instp/ASIdentifierManager/advertisingIdentifier

67. PiOS tracks the use of several pieces of privacy sensitive information, such as the unique device identifier (UDID), location information, address book contents, or browsing history.

68. PiOS was evaluated on a set of 1,407 free iOS apps. 825 apps were downloaded from the Apple app store, and an additional 582 applications were obtained from the alternative Cydia repository.

1. Advertising and statistics libraries

69. App developers frequently add advertising capabilities to their applications. Showing advertisements to the user, creates a stream of revenue for the developer. Similarly, developers often make use of statistics or metrics providers that collect detailed usage statistics of applications.

70. Both these techniques are implemented by the same mechanism. That is, the service provides the developer with a binary library. Furthermore, the developer receives the corresponding header-files and API documentation to make use of this library. Recall that as discussed in Section C.1 the header files containing the function declarations are sufficient for the developer to make use of the functionality implemented in the binary only library. The developer then writes code that uses the provided API and statically links the library into her application.

71. However, beyond the description of the library the developer has no way of assessing what functionality is included in this binary only library. Furthermore, by linking the library into the application these two originally different pieces (i.e., the developers app and the additional library) become a single entity. Therefore, for all intents and purposes, the functionality in the library is now part of the application.

72. This is an unacceptable situation for privacy conscious users. A user that grants location access to a location-aware application, such as a nearby restaurant finder, implicitly grants the whole application access to the location data. With the above mentioned equality of developer authored code and library code, this means that all libraries that are linked into such an application also have unfettered access to the device's location.

[redacted]

74. Furthermore, Apple is aware that marketers and advertisers harvest sensitive user data.

[redacted]

2. PiOS results

76. When evaluating PiOS I observed that 772 applications (i.e., 55%) include at least one of ten popular advertising or statistics libraries. The most prevalent advertising library was Google's AdMob library present in 538 applications, thus covering 38% of the analyzed applications. On each request for a new advertisement, AdMob transmitted the UDID to their back-end servers. Additionally, if the user granted the application access to location, AdMob also transmits geo-location information in these requests. Furthermore, these requests contain a series of values identifying the developer along with other pieces of information. The developer information is necessary at least for billing purposes. The other mentioned libraries showed similar functionality.

77. Note that an application can have a legitimate use for location information (e.g., a nearby restaurant finder). Because developer code is not distinguished from library code, the granted permission automatically extends to the library as well.

78. Until recently, the location permission dialog was generic for all applications. Thus, it was impossible for the user to infer from this dialog for what purpose an application needs location information. While it intuitively makes sense that an application such as a near-

by restaurant finder needs access to location information, the user was left in the dark about all principals (i.e., the application and all included libraries) she's granting access to.

79. Furthermore, because the UDID is persistent for the lifetime of the device and transmitted to these external parties, advertisers and statistics providers can aggregate detailed usage profiles of the devices associated to these UDIDs. Because the UDID is the same for all applications on a device, these external parties gather that data from all applications that include their libraries.

80. The fact that these libraries repeatedly transmit this privacy sensitive data to their servers is unfortunate at best. However, also the code authored by developers accesses and transmits privacy sensitive information. More precisely, 21% of the app store apps (e.g., BibleStories or LoanCalculator) transmit the UDID. Furthermore, 4% of the app store apps were found to transmit location data. Additionally, PiOS identified applications that read and transmit the user's address book in its entirety.

81. Of course, I reported these findings directly to Apple. Unfortunately, however, I was told that Apple is not in charge of privacy issues. Instead I was advised to discuss privacy concerns with the app developer directly.

82. Note that the analysis performed by PiOS is in many aspects conservative. Thus, the numbers reported here are lower bounds as PiOS might not have detected all uses of privacy sensitive data.

E. Resource consumption

83. The practices outlined above have serious impact on user privacy. Moreover, these practices and techniques consume resources on iDevices.

(1) Energy consumption

84. Receiving, processing, and transmitting information of any kind in a mobile device consumes energy. Furthermore, mobile devices are powered by batteries that can only store finite amounts of electric energy.

85. Apple's iDevices are no exception to this rule. Activities, such as Internet browsing, making phone calls, or navigation rely on the iDevices processing, receiving, or transmission capabilities.

86. Thus, the more these activities are performed, the faster the battery is drained and the user has to recharge the device's battery. This is especially true for the resource intensive GPS receiver.

87. Unfortunately, the maximum capacity a rechargeable battery can hold decreases over time. That is, each time the battery is recharged, the capacity decreases. The maximum capacity of a mobile device battery directly translates to the usefulness of that device. Clearly, a mobile device whose battery cannot hold its charge for more than an hour is of limited use to its user. Therefore, a user may wish to exchange the device's battery once the battery's capacity has fallen under a limit the user would deem acceptable.

88. Unfortunately, a regular user cannot replace the batteries contained in iDevices. Instead, Apple offers a battery replacement service at US$ 79 plus shipping [4]. Thus, by conserving energy, the user can extend the time until she has to recharge the device again, therefore prolongating the life of the battery.

89. To this end, Apple provides usage tips to increase the battery life in iPhones [5]. These usage tips include recommendations to only use location services when needed, and minimizing data transmissions to fetch emails or support push notifications.

90. The completion of a single task can involve multiple components of an iDevice. Thus, all these components might require their share of energy to contribute to the completion of that task.

91. For example, a location aware application that embeds advertisements would involve at least the following subsystems on an iDevice. Initially, the code responsible for fetch-

---

[4] http://support.apple.com/kb/index?page=servicefaq&geo=United_States&product=iphone

[5] http://www.apple.com/batteries/iphone.html

ing an advertisement from the ad provider is executed on the device's CPU. This code will use the APIs provided by the CoreLocation framework to determine the current location. If necessary, CoreLocation will interact with the operating system to power up the GPS subsystem to obtain that information. Once the position is known to CoreLocation it will be reported back to the application. Subsequently, this information is transmitted to the ad provider in a network request (commonly this is done via HTTP). This request can be sent via WiFi or the mobile data network depending on how the device is connected to the Internet. The ad provider will chose an ad that should be displayed to the user, and sends that information as response to the request. The application will interpret the response and render the corresponding advertisement on the screen. For this simple use-case at least four different components consumed energy from the battery. The CPU, the GPS subsystem, the WiFi subsystem, and the Screen.

(2) Storage

92. Apple maintains a *crowd-sourced* database that stores location information pertaining to WiFi access points and cell towers. This data is used to provide AGPS (assisted GPS) services. The main advantage of AGPS over regular GPS is that auxiliary coarse-grained location information is used to reduce the *time to first fix* for the GPS receiver.

93. On iDevices this auxiliary information is retrieved from the crowd-sourced database maintained on Apple's servers. Therefore, as the iDevice knows its approximate location already, the time to obtain a fix on the GPS satellites is reduced.

94. This database is populated with location information submitted by the millions of iDevices whose location services are enabled. During regular use, an iDevice caches location information of nearby WiFi access points and cell towers in a local SQLite database stored in the file consolidated.db or in more recent iOS versions cache.db. More specifically, the collected information is stored in a database table called locationharvest. The iDevice will then upload harvested location information to the Apple servers.

95. The data stored in the database table reduces the storage capacity of the iDevice. Thus, reducing the amount of storage the user can make use of for her own intended purposes. Of course, transmitting this data also consumes energy.

96. Additionally, Apple used to back up plain-text (i.e., not-encrypted) copies of this file when backing up an iDevice to the user's Computer. The backup would only be encrypted if the user explicitly chose this non-default setting.

(3) Mobile Data

97. iPhone users commonly have a data plan with their cell-phone provider. Any data that is transmitted consumes bandwidth. Unfortunately, unless the carrier has exceptions in place, all data transmissions count towards the monthly data allowance.

F. Other information

98. My opinion as expressed within this document is based on my extensive research on mobile systems. Furthermore, I considered the following in forming my opinion:

- Documents listed in the concluding Endnotes section G.
- Transcripts of the depositions of Mr. Phillip Shoemaker, Mr. Ronald Huang, Mr. Erik Neuenschwander, and Mr. Guy "Bud" Tribble.
- Plaintiffs' Third Amended Consolidated Class Action Complaint (Document No. 104) and Defendant Apple, Inc.'s Answer (Document No. 113).
- Plaintiffs' Dec. 7, 2012 Disclosure Of Expert Witness Regarding Class Certification.
- Documents produced by Apple in this case.

99. My compensation to be paid for my study and testimony in this case is US$ 400 for consultation and US$ 600 for testimony.

G. Endnotes

[1] Manuel Egele, Christopher Kruegel, Engin Kirda, and Giovanni Vigna. PiOS: Detecting Privacy Leaks in iOS Applications. In *Network and Distributed System Security Symposium, NDSS 2011*, San Diego, CA, USA, 2011.

[2] Josh Hornthal. Privacy & App Review, 2011. Bates:APPLE_1006784.

[3] Cynthia Manning. VIDEOTAPED DEPOSITION OF ERIK NEUENSCHWANDER, November 30, 2012.

[4] Cynthia Manning. VIDEOTAPED DEPOSITION OF PHILLIP B. SHOEMAKER, November 2012.

[5] Henry Mason. Email: Subject: Re: RFC: New tracking identifiers in UIDevice, April 18, 2012. Bates:APPLE_1029837.

[6] Henry Mason. Email: Subject: Re: RFC: New tracking identifiers in UIDevice, April 18, 2012. Bates:APPLE_1029825.

[7] Stephen Moseley. Email: Subject: Re: WSJ privacy story, Dec 19, 2010. Bates:APPLE_0075125.

[8] David Remahl. Device identifiers in iOS Privacy Review, July 2011. Bates:APPLE_1029714.

[9] Josh Shaffer. Email: Subject: Re: Draft API for Identifiers TLF, April 4, 2012. Bates:APPLE_1029826.

[10] Phillip Shoemaker. Email: Subject: Re: WSJ privacy story, Dec 10, 2010. Bates:APPLE_0069455.

[11] Phillip Shoemaker. Email: Subject: Re: UDID Now Verbotten? , 2012. Bates:APPLE_0007935.

[12] Eric Smith. iPhone Applications & Privacy Issues: An Analysis of Application Transmission of iPhone Unique Device Identifiers (UDIDs). http://www.pskl.us/wp/wp-content/uploads/2010/09/iPhone-Applications-Privacy-Issues.pdf, October 2010.

[13] John Montbriand. Email: Subject: Re: Your Apps Are Watching You - The Wall Street Journal., February 9, 2011. Bates:APPLE 1007656.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed on December 17, 2012 at Allegheny County, Pennsylvania.

MANUEL EGELE

ATTACHMENT

Curriculum Vitae

Manuel Egele

1 Personal Information

Name:	Manuel Egele
Address (work):	Carnegie Mellon Univeristy
	Collaborative Innovation Center
	4720 Forbes Avenue
	Pittsburgh, PA, 15213
Email:	megele@cmu.edu
Web:	http://www.iseclab.org/people/maeg
Nationality:	Austria

2 Education

10/2007 - 01/2011

Dr.techn. (Ph.D.) in Computer Science from Vienna University of Technology, Vienna, Austria.
Advisors: Prof. Dr. Christopher Kruegel and Prof. Dr. Engin Kirda

10/2000 - 12/2006

Dipl.-Ing. (M.Sc.) in Computer Science from Vienna University of Technology, Vienna, Austria.
Advisors: Prof. Dr. Christopher Kruegel and Prof. Dr. Engin Kirda

3 Professional Appointments

10/2012 - current

Post-doctoral Researcher at CyLab of Carnegie Mellon University, Pittsburgh, PA, *working with Professor David Brumley*

01/2011 - 10/2012

Post-doctoral Researcher at Department of Computer Science, University of California, Santa Barbara, CA, *working with Professors Giovanni Vigna, Richard Kemmerer, and Christopher Kruegel*

7/2009 - 01/2011

Visiting Researcher at Department of Computer Science, University of California, Santa Barbara, CA, *working with Professors Giovanni Vigna, Richard Kemmerer, and Christopher Kruegel*

10/2008 - 7/2009

Visiting Researcher at Ecole d'Ingénieurs & Centre de Recherche, Eurécom, Sophia Antipolis, France, *working with Professors Engin Kirda and Davide Balzarotti*

10/2007 - 10/2008

Research Assistant at the Intl. Secure Systems Lab (Inst. of Computer Aided Automation), Vienna University of Technology, Vienna, Austria, *working with Professors Engin Kirda and Christopher Kruegel*

4 List of Publications

Journal Publications

[1] G. Stringhini, M. Egele, C. Kruegel, and G. Vigna. Poultry Markets: On the Underground Economy of Twitter Followers. In *ACM SIGCOMM Computer Communication Review - Special October issue SIGCOMM '12*. ACM, October 2012

[2] M. Egele, A. Moser, C. Kruegel, and E. Kirda. PoX: Protecting Users from Malicious Facebook Applications. *Computer Communications*, 35(12):1507 – 1515, 2012

[3] M. Egele, T. Scholte, E. Kirda, and C. Kruegel. A Survey on Automated Dynamic Malware Analysis Techniques and Tools. *ACM Computing Surveys*, 44(2):6:1–6:42, Mar. 2012

[4] M. Egele, C. Kolbitsch, and C. Platzer. Removing web spam links from search engine results. *Journal in Computer Virology*, 7:51–62, February 2011

Conference Publications

[5] M. Egele, G. Stringhini, C. Kruegel, and G. Vigna. Compa: Detecting Compromised Accounts on Social Networks. In *Network and Distributed System Security Symposium, NDSS 2013*, San Diego, CA, USA, 2013. (to appear)

[6] G. Stringhini, M. Egele, A. Zarras, T. Holz, C. Kruegel, and G. Vigna. B@bel: Leveraging Email Delivery for Spam Mitigation. In *Proceedings of the USENIX Security Symposium*, Bellevue, WA, August 2012

[7] L. Davi, A. Dmitrienko, M. Egele, T. Fischer, T. Holz, R. Hund, S. Nürnberger, and A.-R. Sadeghi. MoCFI: A Framework to Mitigate Control-Flow Attacks on Smartphones. In *Network and Distributed System Security Symposium, NDSS 2012*, San Diego, CA, USA, 2012

[8] A. Doupe, M. Egele, B. Caillat, G. Stringhini, G. Yakin, A. Zand, L. Cavedon, and G. Vigna. Hit'em Where it Hurts: A Live Security Exercise on Cyber Situational Awareness. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC 2011)*, Orlando, FL, December 2011

[9] M. Egele, C. Kruegel, E. Kirda, and G. Vigna. PiOS: Detecting Privacy Leaks in iOS Applications. In *Network and Distributed System Security Symposium, NDSS 2011*, San Diego, CA, USA, 2011 *(Distinguished Paper Award)*

[10] N. Childers, B. Boe, L. Cavallaro, L. Cavedon, M. Cova, M. Egele, and G. Vigna. Organizing large scale hacking competitions. In *Proceedings of the 7th International Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA'10)*, pages 132–152, Berlin, Heidelberg, 2010. Springer-Verlag

[11] M. Balduzzi, M. Egele, E. Kirda, D. Balzarotti, and C. Kruegel. A solution for the automated detection of clickjacking attacks. In *ASIACCS '10: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 135–144, New York, NY, USA, 2010

[12] M. Egele, L. Bilge, E. Kirda, and C. Kruegel. CAPTCHA smuggling: hijacking web browsing sessions to create CAPTCHA farms. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1865–1870, New York, NY, USA, 2010

[13] M. Egele, P. Wurzinger, C. Kruegel, and E. Kirda. Defending Browsers against Drive-by Downloads: Mitigating Heap-Spraying Code Injection Attacks. In *Proceedings of the 6th International Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA'09)*, pages 88–106, Milan, Italy, 2009

[14] M. Egele, C. Kruegel, and E. Kirda. Removing web spam links from search engine results. In *18th European Institute for Computer Antivirus Research (EICAR) Conference*, Berlin, Germany, May 2009

[15] H. Yin, D. X. Song, M. Egele, C. Kruegel, and E. Kirda. Panorama: capturing system-wide information flow for malware detection and analysis. In *ACM Conference on Computer and Communications Security (CCS)*, pages 116–127, Alexandria, VA, USA, 2007

[16] M. Egele, C. Kruegel, E. Kirda, H. Yin, and D. X. Song. Dynamic Spyware Analysis. In *Proceedings of the 2007 USENIX Annual Technical Conference*, pages 233–246, Santa Clara, CA, USA, 2007

[17] M. Egele, M. Szydlowski, E. Kirda, and C. Kruegel. Using Static Program Analysis to Aid Intrusion Detection. In *Proceedings of the 3rd International Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA'06)*, pages 17–36, Berlin, Germany, 2006

Workshop Publications

[18] G. Stringhini, M. Egele, C. Kruegel, and G. Vigna. Poultry Markets: On the Underground Economy of Twitter Followers. In *Proceedings of the Workshop on Online Social Network (WOSN)*, Helsinki, Finland, August 2012. ACM

[19] M. Szydlowski, M. Egele, C. Kruegel, and G. Vigna. Challenges for Dynamic Analysis of iOS Applications. In *iNetSec2011 Open Research Problems in Network Security*, Luzerne, Switzerland, 2011

[20] L. Davi, A. Dmitrienko, M. Egele, T. Fischer, R. Hund, S. Nürnberger, A.-R. Sadeghi, and T. Holz. CFI Goes Mobile: Control-Flow Integrity for Smartphones. In *International Workshop on Trustworthy Embedded Devices (TrustED)*, Leuven, Belgium, 2011

[21] M. Egele, A. Moser, C. Kruegel, and E. Kirda. PoX: Protecting Users from Malicious Facebook Applications. In *3rd IEEE International Workshop on SEcurity and SOCial Networking (SESOC)*, Seattle, WA, USA, March 2011

[22] M. Egele, E. Kirda, and C. Kruegel. Mitigating Drive-by Download Attacks: Challenges and Open Problems. In *iNetSec2009 Open Research Problems in Network Security*, Zurich, Switzerland, 2009

Theses

[23] M. Egele. *Protecting Web Clients from Internet Threats*. PhD thesis, Vienna University of Technology, Austria, 2011

[24] M. Egele. Behavior-Based Spyware Detection Using Dynamic Taint Analysis. Master's thesis, Vienna University of Technology, Austria, 2006

5 Invited Talks

1. Mobile Security iOS & Android, *ETH Zürich,* August 2012
2. The State of Mobile Security, *9th Conference on Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA),* July 2012
3. On the Importance of Detecting Compromised Accounts on Social Networks, 23^{rd} *annual Communications and Digital Signal Processing (CDSP) Research Workshop,* April 2012

6 Awards

1. Distinguished paper award for *PiOS: Detecting Privacy Leaks in iOS Applications* at *Network and Distributed System Security Symposium, NDSS 2011*
2. Student travel grant to attend *Usenix Security Symposium, 2010*
3. Student travel grant to attend *AsiaCCS, 2010*

7 Professional Activities

Program Committee Memberships

1. USENIX Workshop on Offensive Technologies (WOOT), 2013
2. Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA), 2012, 2013
3. ASE/IEEE International Conference on Cyber Security, 2012
4. USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), 2012

External Reviewer

5. IEEE Symposium on Security and Privacy, 2010, 2012, 2013
6. ACM Conference on Computer and Communications Security, 2010, 2012
7. International Symposium on Recent Advances in Intrusion Detection, 2011
8. Network and Distributed System Security Symposium (NDSS), 2010
9. International Information Security Conference (SEC), 2010
10. International Workshop on Software Engineering for Secure Systems (SESS), 2009

Journal Reviewer

11. ACM Computing Surveys
12. The Computer Journal
13. IEEE Journal of Internet Computing
14. Elsevier Computers & Security (COSE)
15. Elsevier Journal of Systems and Software (JSS)
16. Springer International Journal of Information Security (IJIS)
17. Elsevier Computer Networks (COMNET)

8 Research Grants

01/2012 - 06/2015

DarkDroid: Exposing the Dark Side of Android Marketplaces *funded by Defense Advanced Research Projects Agency (DARPA); US$ 2.5M; Co-author of the research grant proposal.*

9 Consulting

01/2012

Kramer Levin Naftalis & Frankel LLP, Menlo Park, CA